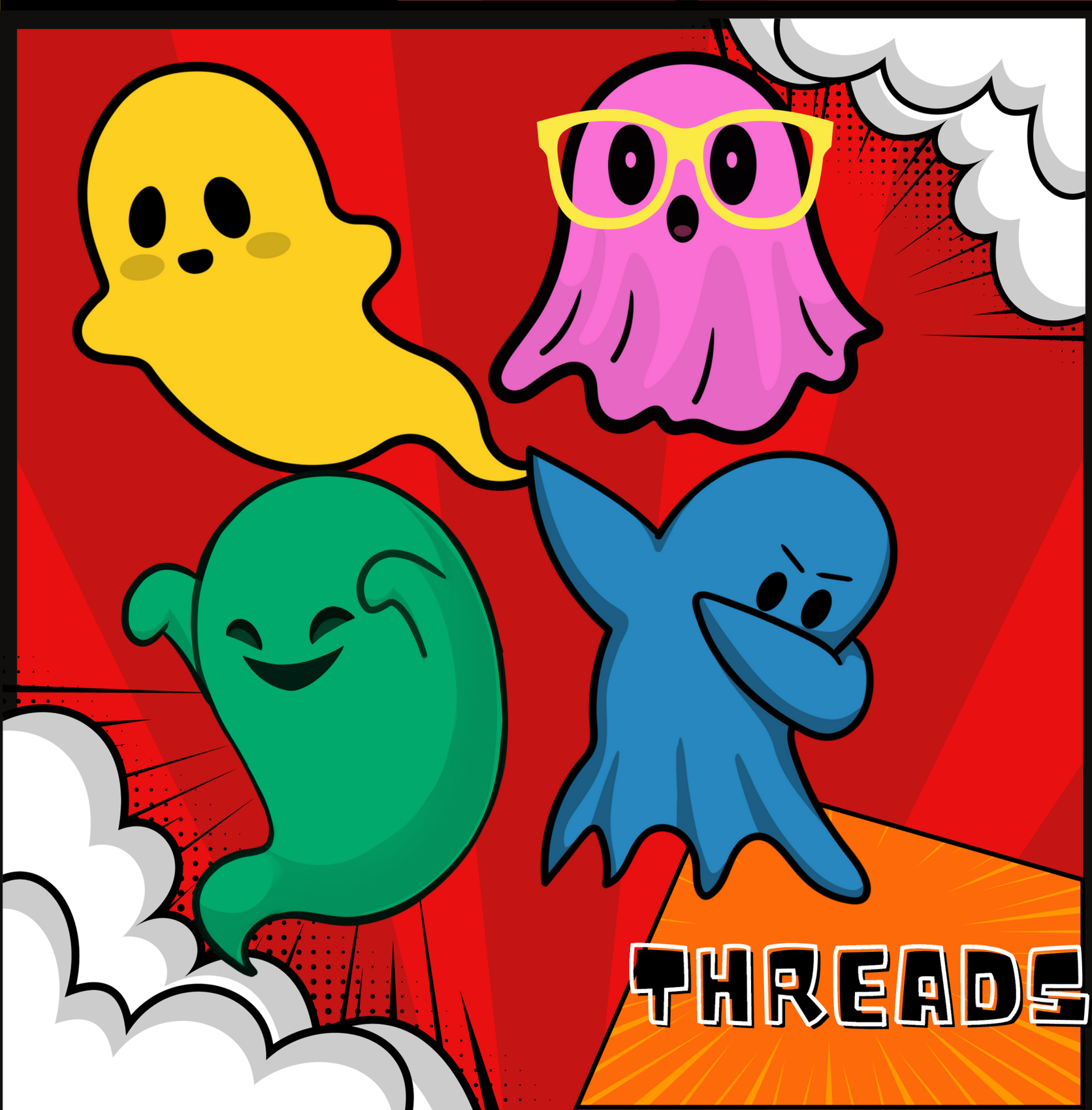
# CHAT VERSE #168

## LAB 01 EDITION

By. Alexander Bañados
ECE 168 WWX

HELLO!

MEET **Serva!** THE OVERWORKED BUT EVER-SMILING CLERK OF CHATVERSE#168—KEEPER OF THE SECRET BOOK, DISTRIBUTOR OF GOSSIP, AND MISTRESS OF THE THREADS!




# THE SECRET BOOK

(JUST AN ATTENDANCE SHEET)

10.158.0.1

10.158.0.2

10.158.0.3

10.158.0.4

"HI-HI! WE'RE THE GHOSTLY THREADS! THINK OF US AS LITTLE PARALLEL ASSISTANTS. EACH OF US GRABS A CLIENT, LISTENS TO THEIR NONSENSE, AND WHISPERS IT BACK TO SERVA. THAT WAY, NOBODY GETS STUCK WAITING—NON-BLOCKING MAGIC, WOOOOOO~"

CHANGE YOUR NAMES USING THE SECRET CODE:

/username

WELCOME TO CHATVERSE#168. YOU NAMES CONFUSE ME!

I GAVE EACH ONE OF YOU A GHOSTLYTHREAD SO YOU DON'T HAVE TO FORM A LINE WAITING TO TELL ME ALL YOUR PACKETS

SCARED OF THE RULES? DON'T WORRY, THE EXIT ISN'T HAUNTED—JUST WHISPER:

/quit

AND YOU'RE OUT!

FROM NOW ON, CALL ME KLAY... AND YOU, MY GHOSTTHREAD, IS NOW ENT!

LET ME WRITE THE SECRET CODE AS WELL SO SERVA WOULD KNOW WHAT I MEAN...

/username Kent

MESSAGES ARE DELIVERED AS PACKETS

# MEANWHILE IN THE SOCKET




YES, KLAY CAN JUST TELL SERVA HIS MESSAGE/COMMAND DIRECTLY. BUT HAVING ENT MEANS HE DOESN'T NEED TO WAIT FOR SOMEBODY TO FINISH TALKING TO SERVA.

KLAY HATES WAITING! KLAY HATES GETTING INTERRUPTED! ENT IS A GREAT HELP!

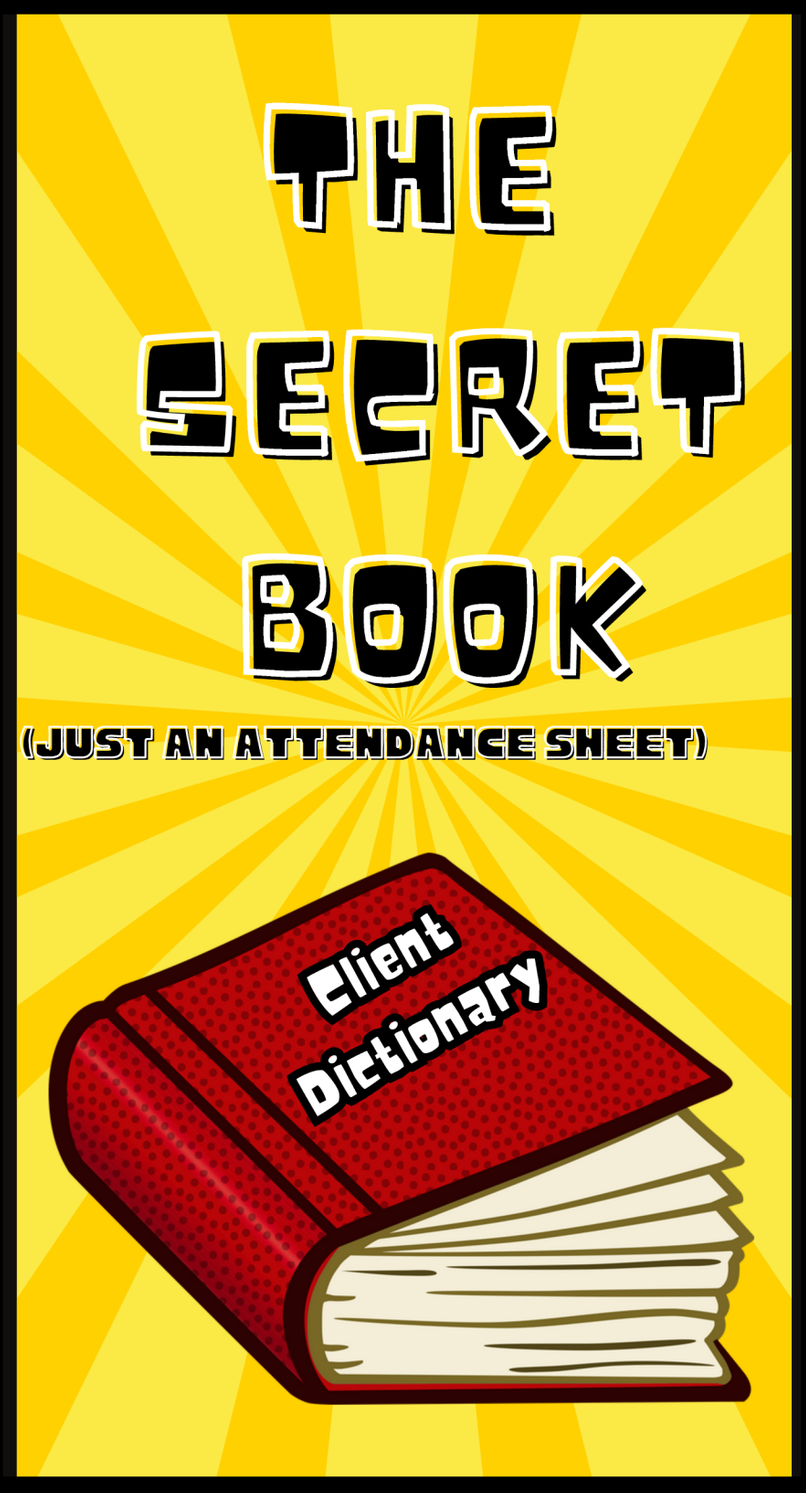
WAIT... DIDNT KENT JUST WROTE HIS MESSAGE ON A PAPER? WHY IS ENT CARRYING AN ENVELOPE NOW... WHAT COULD THIS ENVELOPE MEAN...?

TO BE CONTINUED...

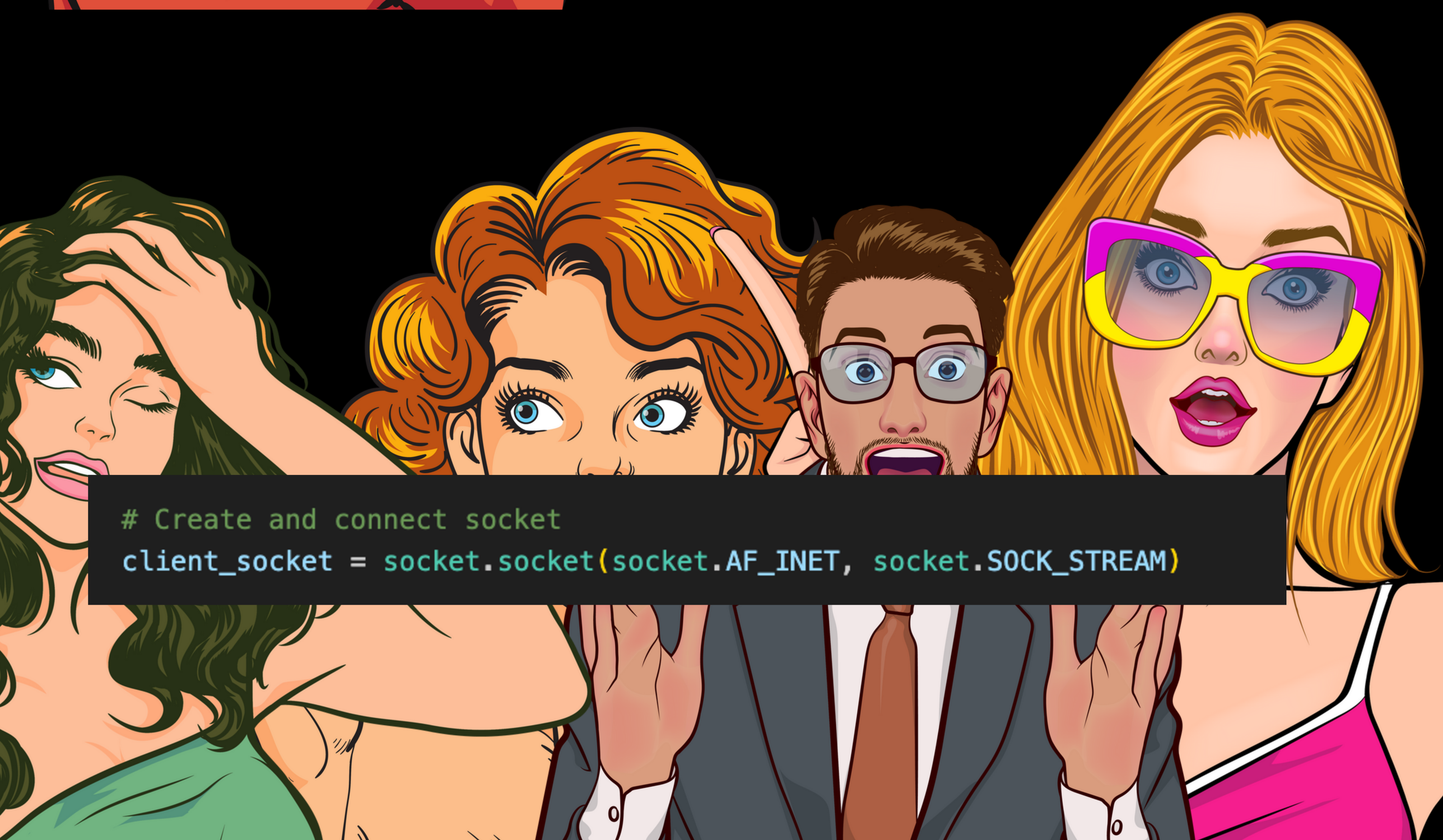
```python
try:
    print(f"Connecting to {host}:{port}...")
    client_socket.connect((host, port))
    print("Connected to chat server!")

    # Start thread to receive messages
    receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
    receive_thread.daemon = True
    receive_thread.start()
```

```python
def receive_messages(client_socket):
    """Thread function to continuously receive messages from server"""
    while True:
        try:
            message = client_socket.recv(1024).decode()
            if not message:
                print("\nConnection lost with server")
                break

            # Clear current input line and print server message
            print(f"\r{' ' * 50}\r{message}")
            print(" -> ", end="", flush=True)

        except (ConnectionResetError, ConnectionAbortedError, OSError):
            print("\nConnection lost with server")
            break
        except UnicodeDecodeError as e:
            print(f"\nError decoding message: {e}")
            break
```

```python
class ChatServer:
    def __init__(self):
        self.clients = {}  # Dictionary to store client connections and usernames
        self.client_counter = 0
```



```python
# Get local IP address
# host = socket.gethostbyname(socket.gethostname())
host = "0.0.0.0"
print(f"Server will start on {host}:{port}")

# Create chat server instance
chat_server = ChatServer()

# Create and configure server socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

try:
    # Bind to the specified address and port
    server_socket.bind((host, port))
    print(f"Server bound to {host}:{port}")
```



```python
# Create and connect socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

SCARED OF THE RULES?
DON'T WORRY, THE EXIT ISN'T
HAUNTED—JUST WHISPER:

/quit

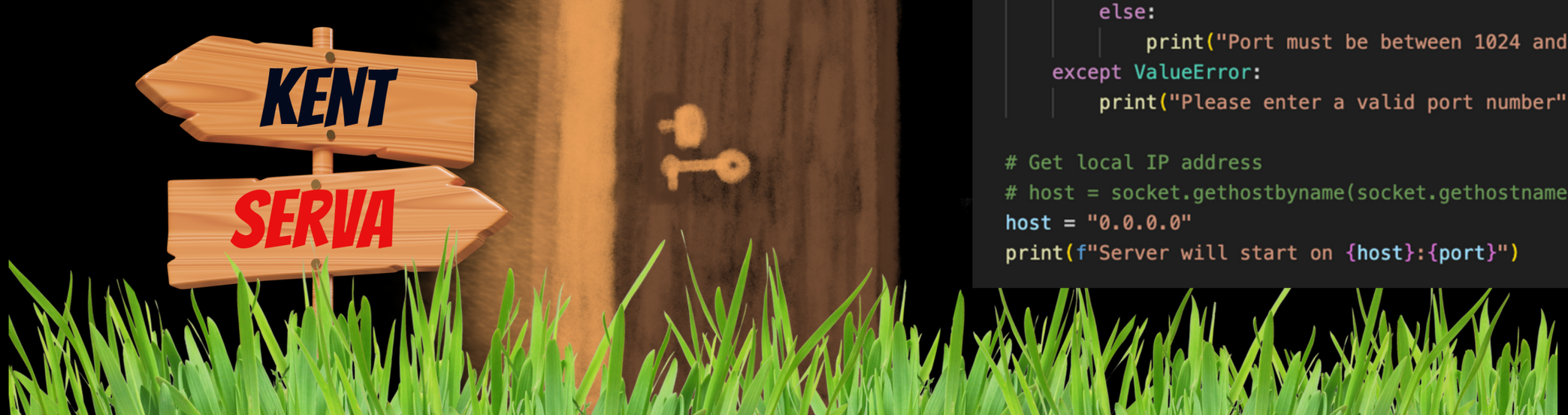
AND YOU'RE OUT!

/username Kent

```python
# Process different types of messages
if data.startswith('/quit'):
    # Client wants to quit
    self.remove_client(conn)
    break

elif data.startswith('/username '):
    # Client wants to change username
    new_username = data[10:].strip()  # Remove '/username ' prefix
    if new_username and new_username != current_username:
        old_username = current_username
        self.clients[conn]['username'] = new_username

        # Notify all clients of username change
        change_message = f"[SERVER] {old_username} changed their username to {new_username}"
        self.broadcast_message(change_message)
        print(f"[UPDATE] Username changed: {old_username} -> {new_username}")
    else:
        error_msg = "[SERVER] Invalid username or username unchanged"
        conn.send(error_msg.encode())

else:
    # Regular chat message - broadcast to all other clients
    chat_message = f"{current_username}: {data}"
    self.broadcast_message(chat_message, conn)
```

```python
def broadcast_message(self, message, sender_conn=None):
    """Broadcast a message to all connected clients except the sender"""
    disconnected_clients = []

    for conn in self.clients:
        if conn != sender_conn:
            try:
                conn.send(message.encode())
            except (ConnectionResetError, BrokenPipeError, OSError):
                # Client disconnected, mark for removal
                disconnected_clients.append(conn)

    # Remove disconnected clients
    for conn in disconnected_clients:
        self.remove_client(conn)
```

```python
while True:
    try:
        host = input("Enter server IP address (or press Enter for localhost): ").strip()
        if not host:
            host = "127.0.0.1"  # localhost

        port = input("Enter server port: ").strip()
        if not port:
            print("Port is required!")
            continue
        port = int(port)
        break
```




```python
while True:
    try:
        port = int(input("Enter the port number to use (1024-65535): "))
        if 1024 <= port <= 65535:
            break
        else:
            print("Port must be between 1024 and 65535")
    except ValueError:
        print("Please enter a valid port number")

# Get local IP address
# host = socket.gethostbyname(socket.gethostname())
host = "0.0.0.0"
print(f"Server will start on {host}:{port}")
```